Modeling Service Level Agreements with Linked USDL Agreement

José María García, Pablo Fernández, Carlos Pedrinaci, Manuel Resinas,
Jorge Cardoso, and Antonio Ruiz-Cortés

Abstract—Nowadays, service trading over the Web is gaining momentum. In this highly dynamic scenario, both providers and consumers need to formalize their contractual and legal relationship, creating service level agreements. Although there exist some proposals that provide models to describe that relationship, they usually only cover technical aspects, not providing explicit semantics to the agreement terms. Furthermore, these models cannot be effectively shared on the Web, since they do not actually follow Web principles. These drawbacks hamper take-up and automatic analysis. In this article, we introduce Linked USDL Agreement, a semantic model to specify, manage and share service level agreement descriptions on the Web. This model is part of the Linked USDL family of ontologies that can describe not only technical but also business related aspects of services, incorporating Web principles. We validate our proposal by describing agreements in computational and non-computational scenarios, namely cloud computing and business process outsourcing services. Moreover, we evaluate the actual coverage and expressiveness of Linked USDL Agreement comparing it with existing models. In order to foster its adoption and effectively manage the service level agreement lifecycle, we present an implemented tool that supports creation, automatic analysis, and publication on the Web of agreement descriptions.

Index Terms—Service level agreements, semantic modeling, business services, quality of services, service trading, semantic analysis, service management

1 Introduction

Although service economy is of utmost importance in developed countries and electronic commerce over the Web has been adopted all over the world, parties still perform service trading manually [1]. Tasks like searching for services, analyzing their characteristics, or customizing a contract including service level agreements are generally carried out via manual means.

The Web of services has been envisioned as a complementary economic infrastructure to traditional, brick-and-mortar services. There are several conceptual models and prototypes proposed towards this vision (see e^{3}Service [2], USDL [3], Linked USDL [1], and cloud computing management [4]) that offer facilities to support service trading over the Web in an open, scalable, and automated manner.

Linked USDL has recently emerged as a versatile, general purpose means to formalize service descriptions including various aspects, such as participants, resources, interactions, and distribution channels. Linked USDL has been designed as a modular and extensible family of ontologies, providing convenient facilities to support modeling, processing, and sharing of service descriptions openly on the Web. However, up to now, Linked USDL does not offer coverage for capturing agreement contracts between participants of a service transaction. Service level agreements (SLAs) are among the most relevant of these contracts. An SLA defines the guaranteed level of a service property (e.g., availability and response time) and consequent actions in case of non-compliance situations, including compensations and liability issues.[1]

In this article we present Linked USDL Agreement, an extension to the Linked USDL family of ontologies that provides domain independent means for describing SLAs. This model offers the necessary facilities for capturing the semantics of those agreements so that heterogeneity and interoperability issues present in current SLAs specifications are avoided. Furthermore, Linked USDL Agreement follows Linked Data principles [5], arising as a fundamental building block for online service trading by facilitating both customers and providers to publish, search for, analyze, reuse, and manage the SLAs involved in any service transaction. Therefore, our proposal appropriately supports the SLA lifecycle when compared to other alternatives [6]. Linked USDL Agreement natively embraces foundational principles of the Web of Data in order to share descriptions, and it is accompanied by a reference implementation that validates its suitability to create and analyze SLAs.

The rest of the article is structured as follows. Section 2 provides an overview of the related work in the field of SLAs and introduces Linked USDL. Then, Section 3 presents the requirements and motivating scenarios that have been used to drive this work. Section 4 thoroughly

• *J.M. García, P. Fernández, M. Resinas, and A. Ruiz-Cortés are with the University of Seville, Sevilla 41004, Spain. E-mail: {josemgarcia, pablofm, resinas, aruiz}@us.es.*

• *C. Pedrinaci is with The Open University, Milton Keynes MK7 6AA, United Kingdom. E-mail: c.pedrinaci@open.ac.uk.*

• *J. Cardoso is with the CISUC/Department of Informatics Engineering, University of Coimbra, Coimbra 3000, Portugal. E-mail: jcardoso@dei.uc.pt.*

1. An example of a traditional paper-based SLA contract can be found at http://www.slatemplate.com/ServiceLevelAgreementTemplate.pdf

describes the Linked USDL Agreement module we have devised. Section 5 evaluates our proposal, while Section 6 showcases the implemented tooling. Finally, Section 7 presents the conclusions and our future work.

2 Related Work

USDL [3] may be, to date, the most comprehensive approach for supporting the description of services for automated processing, covering not only functionality, but also their interfaces, pricing models, SLAs, and legal aspects, among others. Despite its exhaustive support, USDL underestimated the flexibility, extensibility, openness and yet simplicity that such a model should provide [1]. In order to overcome these limitations, Linked USDL constitutes an all encompassing model for describing services, inspired by USDL but following an open, simpler, more extensible, and Web-centric design [1].

2.1 The Linked USDL Family

Linked USDL[2,3] is a family of Web vocabularies or modules building upon the existing results and experience acquired with USDL combined with previous research on business ontologies, Semantic Web Services, and Linked Data, aimed at better supporting trading at Web scale [1]. Linked USDL is grounded on two fundamental principles: i) the use of Linked Data [5] for representing and publishing the descriptions of services and relevant entities, e.g., the involved parties; and ii) the exploitation of formal ontology representation languages, albeit lightweight to preserve scalability, in order to represent services and relevant entities semantics.

The Linked Data principles provide a set of best practices for sharing data effectively on the Web. In a nutshell, these principles dictate that one should:

- Use URIs to name (identify) things.
- Use HTTP URIs so that these things can be looked up (interpreted, "dereferenced").
- Provide useful information about what a name identifies when it is looked up, using open standards such as RDF, SPARQL, etc.
- Refer to other things using their HTTP URI-based names when publishing data on the Web.

Since Linked Data principles were outlined in 2006, there has been an outstanding and increasing uptake initially by academia but soon after by major companies. In fact, currently Linked Data principles are regarded as the best means for sharing data online. Among their benefits, Linked Data principles promote and support reuse which helps to reduce the data modeling overhead (e.g., by reusing conceptual models and existing data sets), and in turn helps create a Web of interlinked data ready to be processed automatically by machines.

Linked USDL is mostly modeled using RDF/RDFS constructs with a fairly limited inclusion of abstract concepts so as to attain a model that is simple enough for its use by humans and machines on a Web scale. Linked USDL builds upon several complementary networked vocabularies that provide good coverage of necessary aspects (e.g., modeling temporal aspects) and are widely used on the Web for capturing their particular domains. Among the main vocabularies reused it is worth noting DC Terms, Time Ontology, Minimal Service Model, GoodRelations, and Schema.org among others.

Linked USDL was designed with modularity in mind in order to reduce the overall complexity of service modeling by enabling providers to only use the modules needed. Currently, six modules exist with different degrees of maturity, Linked USDL Core being the unifying foundational one. Among other aspects Linked USDL Core defines the concept `Service` which captures services that can be provided by a given `BusinessEntity`. These services are offered to potential customers through `ServiceOfferings` that establish concrete offering conditions such as additional added-value services, concrete pricing, etc. Another important aspect covered by Linked USDL Core is the involvement of different actors within concrete value chains. In particular, Linked USDL Core defines the concept `EntityInvolvement` which allows to capture a ternary relationship expressing that a `BusinessEntity` is involved in a certain `Service` with a particular `BusinessRole`. This allows for instance capturing things such as the fact that "AXA is the service provider of a basic life insurance offering". The reader is referred to [1] for a detailed description of Linked USDL Core.

Other aspects covered by the Linked USDL family of vocabularies also include a range of SKOS categorizations (e.g., BusinessRoles, InteractionRoles), a Pricing module, and the Linked USDL Agreement module presented in this paper and previously introduced in [7]. There also exist further modules that have been developed only as a proof of concept, namely `usdl-privacy`, `usdl-sec`, and `usdl-ipr`, aiming at describing the main privacy, security, and usage rights properties of a service, correspondingly.

Besides this set of vocabularies there are related initiatives by 3rd parties. This includes for example, Linked USDL4EDU focussed on domain specific services—from education. Another stream of related research is Linked Service System USDL (LSS USDL[4]) [8] which provides modeling constructs to capture the concepts and relationships of a service system.

While covering all existing related models is outside the scope of this paper, it is worth highlighting the wide spread coverage already attained by existing models both in terms of specific issues related to service trading (e.g., pricing, agreements), as well as in terms of domain specific extensions (e.g., education). This highlights to a certain extent the versatility of the approach followed by Linked USDL towards enabling the emergence of a rich family of vocabularies for service trading on the Web. The interested reader can see a more detailed overview of the Linked USDL family in [9].

2.2 Service Level Agreement Models

Although Linked USDL Core provides essential facilities for describing and managing services, given the wide range of aspects that are relevant to service trading, it was purposely designed to support and promote the use of topic (e.g., security) and domain-specific (e.g., logistics) extensions to accommodate the many needs one is likely to encounter. In

2. http://www.linked-usdl.org/
3. http://github.com/linked-usdl/
4. https://w3id.org/lss-usdl/v2

particular, relying on Linked Data standards and semantic representations allows to seamlessly create and attach dedicated extensions, or modules as we call them, to Linked USDL Core on an on-demand and distributed basis.

SLA management is one of those aspects for which a specific extension is needed. Some preliminary work has been done towards such an extension that transforms Linked USDL Business Policies to WS-Agreement [10]. A subset of the original WS-Agreement model but extended with ad-hoc constructors is obtained via this transformation, though it does not cover the compensation elements that are supported in our proposal. Marquezan et al. [11] also extend Linked USDL with a Transport and Logistics SLA Vocabulary. In contrast to our proposed Linked USDL Agreement module, they devised a domain-specific extension. Furthermore, it does not support expressing common terms of existing SLAs such as penalties. Nevertheless, both proposals bring out the clear need for an extension to Linked USDL in order to describe domain independent SLAs.

Apart from USDL, there are several languages or models to specify SLAs in the literature (cf. a comparative analysis in [6]). The most prominent industrial approaches are WSLA [12] and WS-Agreement [13], respectively introduced in 2001 and 2005 by IBM and the Global Grid Forum. The latter constitutes an evolution of the former, and it provides a specification framework that offers extension mechanisms to create fully-fledged SLA languages (cf. [14]). Even though there exist various approaches, most assume that an underlying WSDL description is available. However, in the case of automated services, the use of WSDL services has de-facto been deprecated in favor of Web APIs, and in the case of manual, real-world services there most often does not exist any software endpoint to support automated interactions. In fact, those approaches essentially target software-based (i.e., computational) services which only represent a small, although important portion of the service economy, leaving many other non-computational service activities (e.g., insurance, eLearning, etc.) with poor coverage and support if any (cf. Section 3.1).

In turn, Linked USDL Agreement provides facilities to specify SLAs for any kind of service regardless of its domain and its computational or non-computational nature. Furthermore, our proposal enables the automation of the SLA lifecycle by using formal semantics to define our SLA model. A more detailed comparison with existing SLA languages is discussed in Section 5.3.

3 Requirements and Use Cases

In order to devise an agreement module within the Linked USDL family, we have first identified current challenges present in the SLA research area, which are illustrated with two motivating scenarios from two different domains, namely cloud computing and business process outsourcing services. Analyzing those scenarios, we enumerate a list of competency questions [15] that drive the design of our proposal.

3.1 Challenges on Service Level Agreement

Since the introduction of SLA languages like WSLA and WS-Agreement, important technological developments have been made in the field of services, such as the emergence of cloud services and Web APIs, which have substantially changed the interaction with computational services and, hence, their SLAs. Thus, there are specific challenges that need to be tackled in the SLAs field, which are discussed in the following.

3.1.1 Shared Meaning of Content

Providers and customers need to speak the same "language" for achieving an effective service trading. Therefore, service descriptions have to be specified on an agreed upon format or schema (*shared representation schema*), while being expressed in mutually understandable terms and concepts (*shared meaning of content*). Existing SLA languages only focus on the first requirement. Notably, since they are based on XML, these approaches do not benefit from the inferential capabilities inherent to semantic representations that truly enable the creation of dedicated extensions or refinements of the core concepts shared by the specification (e.g., adding new service types, or kinds of actors involved in a service) while ensuring that the semantics of these extensions are well understood. With XML, extensions are essentially new 'keywords' that are largely unrelated to all pre-existing ones therefore losing the original semantics of the model and ensuring solely a shared representation of data, not its meaning.

In turn, Linked Data was purposely devised to support the publication, search, and interpretation of both schemas and content in a machine understandable form over the Web. For instance, `itil:{processes, roles, glossary}`[5] vocabularies specify more than 600 IT related terms, which can be used within contracts in order to unambiguously share their contents semantics. Section 4 shows the application of Linked Data to specify SLA documents.

3.1.2 Non-Computational Services Support

Services in the real world are above all business activities that may be provided automatically (e.g., cloud computing services). However, more often than not, they are manual activities (e.g., insurance, consultancy, etc.). In this broader context being able to capture, process, and reason about real-world (i.e., non-computational) service offerings, value chains, agreements and guarantees is paramount. Our proposed agreement model accounts for both computational (see Section 5.1) and non-computational services (see Section 5.2).

3.1.3 Open, Web-Based Solution

To effectively share and process SLA descriptions over the Web while promoting take-up, the technological approach should allow anybody to openly publish, search for and exploit such descriptions, but it should also support extensions to address unexpected needs and use cases. In contrast to previous solutions, our proposal embraces Web principles and technologies to address interoperability and scalability issues. The aforementioned limitations for sharing meaning, but also the document-driven nature of XML significantly hampers the organic, distributed and opportunistic growth

5. http://w3id.org/itil/{processes, roles, glossary}

TABLE 1
Amazon EC2 SLA Excerpt

Monthly Uptime Percentage	Service Credit Percentage
Less than 99.95% but equal to or greater than 99.0%	10%
Less than 99.0%	30%

in descriptions and extensions as necessary to enable an open and Web-scale service trading.

In particular, the use of Linked Data principles, which represent best practices for sharing data and its semantics openly on the Web, ensures that new service descriptions, new dedicated extensions to cover specific aspects (e.g., SLAs), or simply valuable 3rd party defined general descriptions (e.g., companies descriptions), can directly be (re)used and integrated as they are found and on an as-required basis. Section 5.3 compares our solution to other SLA approaches.

3.1.4 SLA Lifecycle Automation

The SLA lifecycle comprises not only the essential negotiation and creation of SLAs, but also validity checking, conformance and monitoring of contracts to detect conflicts and violations. These activities are usually carried out manually, resulting in expenses and errors. Automated software tools for SLA documents are necessary to carry them out efficiently. Section 6 demonstrates how this automation can be achieved.

3.2 Use Cases

We have chosen two use cases from different domains in order to cover a broad spectrum of competency questions for SLAs, namely a cloud computing services use case and a business process outsourcing services use case.

3.2.1 Cloud Computing Services Use Case

Cloud computing has turned out as a cost-effective and efficient paradigm for on-demand provisioning of computing services. Instead of hosting a large number of computing resources on site, businesses can dynamically use external services that provide them, decreasing the maintenance and operating costs, while obtaining a highly scalable infrastructure [16]. Usually, cloud computing solutions focus on four different layers: hardware, infrastructure, platform and application. Vendors offer services associated to different layers, depending on the users requirements. For instance, *Infrastructure as a Service* (IaaS) consist of a service providing resources at the infrastructure layer, such as servers and virtual machines. Some IaaS providers are Amazon EC2,[6] Microsoft Azure,[7] and Google Cloud Platform.[8]

In this scenario, where several parties interact using services to subcontract computing resources, SLAs governing the parties relationship need to be formalized. However, providers mostly offer natural language descriptions of SLAs in their websites. For instance, Table 1 shows a typical service commitment from the SLA of Amazon EC2.[9] This information was intended to be interpreted by humans and not by software.

TABLE 2
BPO Service SLA Excerpt

Indicator	Target	Penalty
Percentage of very high priority issues resolved in less than 2 hours since the issue was assigned	$\geq 90\%$	$\frac{90-IO\_01}{10} \times 30 \times P_{max}$
Average resolution time of high-priority issues	≤ 6h	$\frac{IO\_05-6}{1} \times 20 \times P_{max}$
Percentage of issues that are re-opened	$\leq 1\%$	$\frac{IO\_08-1}{4} \times 20 \times P_{max}$

3.2.2 Business Process Outsourcing (BPO) Services Use Case

Business Process Outsourcing services are non-computational services that allow the customer a partial or full outsource of a business process to the service provider. These business processes typically include logistics processes, supply-chain processes, or IT delivery processes. Like computational services, the execution of these services are regulated by SLAs and compensations are established in case the service level values guaranteed by the SLA are not met.

An example of BPO service of these characteristics is the maintenance of the human-resources information systems and the web and intranet of the Andalusian Health System.[10] In this scenario, the Andalusian Health System outsources the business processes in charge of the strategy, design, transition and operation of their human-resources information systems, their Web site and their intranet. This outsourcing is done via a public tendering process. As a part of the tender request documents that are prepared by the Andalusian Health System, an SLA is defined to make sure that the BPO service is provided within acceptable service level parameters. For example, Table 2 shows an excerpt of the SLA designed for the aforementioned service, where each indicator (e.g., *IO_01*) is defined along with its target value and the corresponding percentage of penalty to apply to incurred costs in case of violations (P_{max} being the maximum percentage of penalty as specified in the SLA).

Like in the cloud computing services use case, this SLA is described in natural language. However, having this information in a way that can be machine processable could bring many advantages, such as the automated evaluation of SLA improvements made by tenders during the tendering process, or the automated computation of the service level values during service execution. Furthermore, a precise definition of the metrics used in the SLA is crucial to avoid different interpretations between service provider and consumer.

3.2.3 Competency Questions

Analyzing the described use cases and usual contents of SLA documents, as is common practice for ontology

6. http://aws.amazon.com/ec2
7. http://azure.microsoft.com
8. https://cloud.google.com
9. http://aws.amazon.com/ec2/sla/
10. http://www.juntadeandalucia.es/contratacion/ContractNoticeDetail.action?code=2015-0000008807

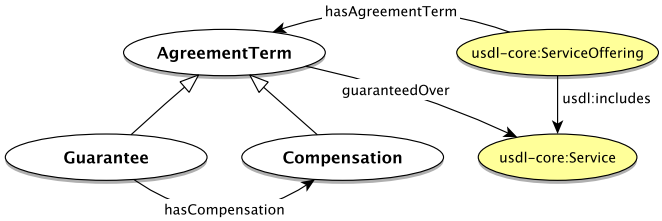

Fig. 1. Linked USDL Agreement main concepts.

modeling, we have defined a series of competency questions that operationalize the requirements that a semantic vocabulary useful for the SLA lifecycle management should have. In a nutshell, competency questions [15] are natural language sentences that define the type of questions that people expect an ontology to answer. The main ones are:

- Q1: Which functionality and quality of service levels does a service deliver?
- Q2: Which particular properties of a service are guaranteed to have certain values?
- Q3: Which compensations are offered if the guaranteed value of a property is not honored?
- Q4: Who is the responsible party for enforcing the guaranteed service level values?
- Q5: Who is the responsible party for monitoring and computing the guaranteed values?
- Q6: During which period of time a guarantee is offered?
- Q7: How are current values of a service property computed?

Effectively answering these competency questions is the main requirement we have taken into account when designing our model. Additionally, we impose requirements regarding scalability and its exploitation on the Web. Therefore, we have to reuse existing Web standards and technologies that facilitates the publication and management of SLA documents. Finally, our proposal is informed by major contributions on SLA specification like WS-Agreement [13].

4 Semantic Modeling of Service Level Agreement

Driven by the identified challenges and the previously discussed competency questions, we devised an extension to Linked USDL family of vocabularies called Linked USDL Agreement. This agreement module is publicly available in GitHub,[11] including the representation of the discussed use cases.

4.1 Design Decisions

In order to address the challenges enumerated in Section 3.1, our proposal uses formal ontology representation languages to deal with the structural and semantic heterogeneity affecting SLAs. Therefore, as prescribed by Linked USDL [1], we have adopted Linked Data principles [5], so that our model can be used to share and interlink service agreements over the Web.

The design of our model using Linked Data facilitates reusing related models, datasets, and existing tools. Therefore, we have taken advantage of vocabularies and models already used in Linked USDL, building upon their integration with respect to specific needs of SLA modeling, such as validity intervals or service properties. Nevertheless, we support reusing additional vocabularies to extend the expressiveness and improve the automation of the SLA lifecycle with convenient tools. Successfully answering competency questions were the key driver of our model design, as well as enabling the complete description of both use cases, as discussed in Section 5.

11. https://github.com/linked-usdl/usdl-agreement

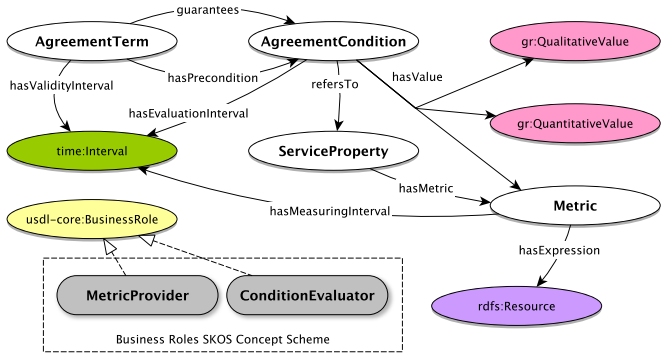

Fig. 2. Agreement conditions and related concepts.

4.2 Linked USDL Agreement Module

Fig. 1 presents the core concepts of our agreement model. Essentially, an agreement comprises a set of terms stating the particular conditions that are guaranteed, and eventually the compensations that may be offered in case a violation of the SLA terms occurs.

Thus, an `AgreementTerm` denotes each term contained in an SLA. A concrete `ServiceOffering` can be associated with several instances of `AgreementTerms`, which represent the complete SLA offered along that offering. Our model differentiates two subtypes of terms that may appear in an SLA, namely *guarantees* and *compensations*.

On the one hand, a `Guarantee` captures an SLA term that *guarantees* the fulfillment of certain conditions over a service property. Other SLA models refer to this concept as a Service Level Objective (SLO). For instance, a *Guarantee* could state that "Amazon guarantees that the monthly uptime of its EC2 service will be at least 99.95 percent".

On the other hand, a `Compensation` describes an alternative term that will be guaranteed in place of the original `Guarantee` term, which is linked to the compensation via the `hasCompensation` property, if it is violated, e.g., "a service credit of 10 percent will be issued if the monthly uptime is less than 99.95 percent but equal to or greater than 99.0 percent". In this example there is also a *precondition* regarding the monthly uptime.

Agreement conditions are further described in Fig. 2. An `AgreementCondition` specifies a constraint or axiom that can be checked within the terms of an SLA. Conditions are used in our model to define the condition that is fulfilled by an agreement term (via the `guarantees` property of an `AgreementTerm`), and to state any precondition that has to be met before guaranteeing the agreement term (via the `hasPrecondition` property). Both types of conditions are usually applied to a concrete service property (associated via the `refersTo` property), constraining their valid *values* as defined by the condition. The part of the

previous guarantee term example stating "the monthly uptime will be at least 99.95 percent" is modeled using `AgreementConditions`.

A `ServiceProperty` is actually a convenience class to simplify the modeling effort of Linked USDL Agreement. It represents either a *qualitative* (e.g., *region state*) or a *quantitative* (e.g., *monthly uptime percentage*) service property, as defined in GoodRelations vocabulary [17]. Agreement conditions can refer to either type of service property.

Service properties may have a specific measurement method, instead of a fixed value. A `Metric` precisely defines that method, usually by a mathematical expression (associated using `hasExpression`) that has to be evaluated in order to obtain the value of a concrete property. For instance, the Amazon EC2 SLA describes that "Monthly Uptime Percentage is calculated by subtracting from 100 percent the percentage of minutes during the month in which Amazon EC2 (. . .) was in the state of Region Unavailable."

Our model provides some pre-defined constructs for incorporating common axiom types in SLAs, including fixed values (as in the previous *compensation* example), intervals, minimums, and maximums. Nevertheless, arbitrary axioms using domain-specific conditional languages may be also described through `rdf:value` and using additional standards like rule languages (e.g., RIF[12]) or even SPARQL queries. The pre-defined axiom types are the following:

- `GuaranteedValue`. A specific agreement condition that checks if the current value of a referred quantitative or qualitative service property is one of the values associated with `hasValue` property of the condition.
- `BetweenGuaranteedValue`. A specific agreement condition that checks if the current value of a referred quantitative service property is contained in the interval defined by the values associated with `hasValue` property of the condition.
- `MinGuaranteedValue`. A specific agreement condition that checks if the current value of a referred quantitative service property is at least the value associated with `hasValue` property of the condition.
- `MaxGuaranteedValue`. A specific agreement condition that checks if the current value of a referred quantitative service property is at most the value associated with `hasValue` property of the condition.

Note that the values of the referred service property that will be checked to evaluate an agreement condition (captured by `hasValue` property) can be either an explicit qualitative or quantitative value (depending on the nature of the referred property), or a metric definition to compute the actual value in runtime.

Concerning the evaluation and measurement of conditions and metrics in the context of an SLA, our model support the specification of time intervals to specify how often an agreement condition is evaluated (using the `hasEvaluationInterval` property), the frequency of metric computation (via `hasMeasuringInterval`) and the validity period of each agreement term (with the `hasValidityInterval` property). Furthermore, Linked USDL

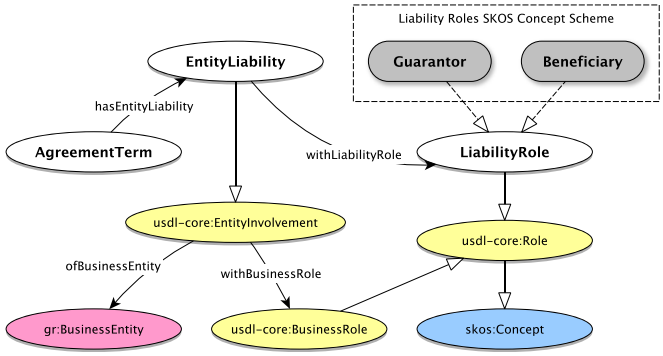

Fig. 3. Business entities liability description.

Agreement extends the SKOS[13] taxonomy of business roles in Linked USDL Core to identify which particular business entity participating in the context of a concrete SLA is responsible for providing metric measurements (`MetricProvider`) and who will evaluate and monitor agreement conditions (`ConditionEvaluator`), since these roles are usually defined in SLAs.

Regarding business entities involvement, Linked USDL Agreement extends the facilities included in Linked USDL Core introduced in Section 2.1. Fig. 3 showcases this extension, resulting in the addition of the `EntityLiability` concept. It captures the *liability role* that an involved *business entity* has in a given term, i.e., its responsibility. An agreement term is associated with an entity liability using the `hasEntityLiability` property, while the entity liability is a ternary relationship extending the `EntityInvolvement` concept relating a `BusinessEntity` (via `ofBusinessEntity`), which is involved in the agreement playing a particular `BusinessRole` (via `withBusinessRole`), with its `LiabilityRole` (via `withLiabilityRole`).

For instance, a service provider can act as a *guarantor* of a guarantee, i.e., being the responsible of its fulfillment. In turn, consumers can play a *beneficiary* role in a compensation since they will benefit from it. Indeed, depending on the specific situation, these roles may be interchanged. Linked USDL Agreement defines a simple SKOS taxonomy of liability roles, which only includes the already discussed `Guarantor` and `Beneficiary` roles. This taxonomy may be extended for specific use cases, when needed.

4.3 Reused Vocabularies

In addition to the previously described main concepts, Linked USDL Agreement rely on several external vocabularies, following Linked Data recommendations. First, as an extension of Linked USDL, our model builds upon Linked USDL Core. Fig. 1 shows that the property `hasAgreementTerm` relates a `ServiceOffering` with its corresponding agreement terms. In turn, each guarantee term is applied to a particular `Service` instance included in the offering, using the `guaranteedOver` property.

GoodRelations [17] is reused by the Linked USDL Core module, and hence we also rely on some concepts from that vocabulary. GoodRelations captures concepts related to commercial activities, such as business entities, products and services. In particular, it defines qualitative and quantitative

12. http://www.w3.org/TR/rif-overview/

13. http://www.w3.org/2004/02/skos/core

```
:amazonEC2ServiceOfferingM1LargeInstanceType
  a usdl-core:ServiceOffering ;
  usdl-agreement:hasAgreementTerm :ec2ServiceCommitment ;
  usdl-core:hasEntityInvolvement
    :involv_customer , :involv_Amazon ;
  usdl-core:includes :ec2M1LargeInstanceType .
```

Listing 1. Service offering linked to the service agreement in Turtle notation.

properties applicable to products or services. Linked USDL Agreement reuse these properties by means of the `ServiceProperty` convenience class, since agreement conditions refer to them. Correspondingly, using the `hasValue` property, a condition specification may use instances of `gr:QuantitativeValue` or `gr:QualitativeValue`.

Temporal properties of SLAs are described in Linked USDL Agreement using the Time Ontology.[14] Thus, agreement conditions can specify evaluation intervals, metrics can be computed in measuring intervals, and SLA terms can restrict their validity period [18]. Concerning metrics definition, we do not impose a particular vocabulary to specify metric expressions, but we recommend the integration with Quantities, Units, Dimensions and Data Types Ontologies (QUDT[15]) for describing and converting units of measurement, or SPIN[16] for defining and computing metric expressions using embedded SPARQL queries.

In order to include general purpose metadata about the vocabulary itself, including provenance and evolution, we also integrate Dublin Core,[17] VANN[18] and Friend of a Friend (FOAF)[19] vocabularies. Finally, as with role schemes defined in Linked USDL Core, we rely on the SKOS vocabulary to create the classification scheme for liability roles.

5 Evaluation

In this section, we evaluate to what extent Linked USDL Agreement address the challenges and the requirements enumerated in Section 3, validating our model using the introduced cloud computing and BPO use cases. We mainly use these real-world scenarios and the set of requirements and competency questions as the reference framework to evaluate our proposed model [19]. An additional real-world validation scenario is discussed in [7]. Furthermore, we compare the SLA description coverage with other current proposals using the framework proposed in [6].

Regarding competency questions, we verified that Linked USDL Agreement model can properly answer our identified competency questions. In order to perform this verification, we translated each competency question into a generic SPARQL query (such as the presented in Listing 2), and executed them over our use cases. Nevertheless, these queries can also be executed over any Linked USDL Agreement description, since they are already integrated in our tooling (see Section 6). In the following we discuss these validation results, introducing the answers to competency questions.

14. http://www.w3.org/TR/owl-time
15. http://qudt.org/
16. http://spinrdf.org/spin.html
17. http://purl.org/dc/terms/
18. http://purl.org/vocab/vann/
19. http://xmlns.com/foaf/0.1/

```
SELECT ?prop WHERE {
  :amazonEC2ServiceOffering
     usdl-agreement:hasAgreementTerm ?term .
  ?term usdl-agreement:guarantees ?conditions .
  ?conditions usdl-agreement:refersTo ?prop }
```

Listing 2. Obtaining service properties relevant to the agreement.

5.1 Cloud Computing Service Agreement

To validate our proposal, we checked its suitability to fully describe the SLA accompanying the EC2 cloud computing service provided by Amazon.[20]

The Amazon EC2 SLA document defines service guarantees with respect to the monthly uptime percentage, which has a guaranteed value for all the EC2 infrastructure. In case the guarantees are not fulfilled, Amazon issues a service credit for the next billing cycle. In this scenario, our model effectively associates SLA guarantee terms with the description of the service offering, which includes a service description that specify its functionality (`:ec2M1LargeInstanceType` in Listing 1), answering the competency question Q1, related to both the quality of service described in the agreement term (`:ec2ServiceCommitment` in Listing 1) and its already mentioned functionality.

SLA terms refer to service properties of Amazon EC2 that are guaranteed. Following our design discussed in Section 4, they can be modeled using the GoodRelations properties schema. Therefore, the competency question regarding service properties (Q2) can be answered by querying the knowledge model about the referenced properties, as in the SPARQL query shown in Listing 2.

The key portion of the EC2 agreement document is the guaranteed condition over the monthly uptime percentage. Listing 3 shows the instantiation of that service commitment. First, we specify the concrete service, included in the previous offering, over which the guarantee term is applied. Liability roles are also described, so that the liability of the different entities involved in the agreement is clear (in our example, an abstract customer and the provider, i.e., Amazon), consequently answering competency question Q4. These entities may also participate in the SLA under additional roles, such as metric provider or condition evaluator, which provide an answer to the competency question Q5. In this use case we define these roles globally, hence they are included in the entity involvement already stated in Listing 1.

Second, we define the guaranteed condition as the minimum value that the `:monthlyUptimePercentage` property can take. Note that the metric definition that computes that property should be also included. In our case, we rely on external vocabularies and tools to properly answer competency question Q7. Third, validity intervals restricting when the term is enforced and evaluation intervals for the monitorization of the agreement are described using the Time ontology intervals, covering question Q6.

Finally, compensation terms model palliative actions that will be taken if guarantee terms are violated, hence answering competency question Q3. Amazon EC2 SLA defines two

20. The complete description of the use case can be found at https://github.com/linked-usdl/usdl-agreement/tree/master/UseCases/AmazonEC2

```
:ec2ServiceCommitment a usdl-agreement:Guarantee ;
   usdl-agreement:guaranteedOver
      :ec2M1LargeInstanceType ;
   usdl-agreement:hasEntityLiability
      :liab_customer , :liab_Amazon ;
   usdl-agreement:guarantees [
      a usdl-agreement:MinGuaranteedValue ;
      qudt:unit
         <http://qudt.org/vocab/unit#Percent> ;
      usdl-agreement:hasEvaluationInterval
         :monthlyInterval ;
      usdl-agreement:hasValue [
         a gr:QuantitativeValueFloat ;
         gr:hasValueFloat "99.95"^^xsd:float ] ;
      usdl-agreement:refersTo
         :monthlyUptimePercentage ] ;
   usdl-agreement:hasCompensation
      :ec2ServiceCredit30, :ec2ServiceCredit10;
   usdl-agreement:hasValidityInterval
      :monthlyInterval .
```

Listing 3. Agreement terms.

compensation layers depending on the actual amount of monthly uptime percentage, as shown in Section 3.2.1. We model these layers associating preconditions to compensation terms. Listing 4 shows one of the compensation levels described in the Amazon EC2 SLA.

5.2 Business Process Outsourcing Service Agreement

We also tested the suitability of our vocabulary to describe the SLA of the BPO service we described in Section 3.2.2, being able to answer the competency questions in a non-computational scenario, too. This SLA contains a number of indicators for each of the four business processes related with the strategy, design, transition and operation of the human-resources information systems, web and intranet of the Andalusian Health System.[21] For each indicator, both a target value and a penalty that applies if the target value is not met are defined. Furthermore, this use case introduces two particularities concerning penalties that were not present in the previous use case. There is an exclusion interval for the application of penalties and there are limits to monthly penalty costs and total penalty costs defined as a percentage of the monthly service cost and total service cost, respectively.

Following the same approach as in the previous use case, we model each of the four processes as sub-services of the BPO service offering and associate the guarantees of the SLA to each of the sub-services to which they apply.

These guarantees model the relationship between indicators, targets, and penalties. Indicators refer to properties of the service that are guaranteed, which are modeled using the properties definition from GoodRelations. Furthermore, a metric is included to describe how each property is computed. In this case we use a SPARQL query to precisely define the metric in terms of issues, response time, and priority as (see Listing 5). Therefore, a SPARQL engine can be directly used to compute the value of the corresponding indicator.

Targets are modeled using agreement conditions by extending `usdl-agreement:MinGuaranteedValue` or `usdl-agreement:MaxGuaranteedValue`. Finally,

21. The description of the operational services part of this use case can be found at https://github.com/linked-usdl/usdl-agreement/tree/master/UseCases/AndalusianHealthService

```
:ec2ServiceCredit10 a usdl-agreement:Compensation ;
  usdl-agreement:hasEntityILiability
    :liab_customer , :liab_Amazon ;
  usdl-agreement:guarantees [
    a usdl-agreement:GuaranteedValue ;
    qudt:unit <http://qudt.org/vocab/unit#Percent> ;
    usdl-agreement:hasValue [
      a gr:QuantitativeValueFloat ;
      gr:hasValueFloat "10"^^xsd:float ] ;
    usdl-agreement:refersTo :ServiceCreditPercentage ] ;
  usdl-agreement:hasPrecondition [
    a usdl-agreement:MinGuaranteedValue ;
    qudt:unit <http://qudt.org/vocab/unit#Percent> ;
    usdl-agreement:hasValue [
      a gr:QuantitativeValueFloat ;
      usdl-agreement:hasValueFloat "99"^^xsd:float ] ;
    usdl-agreement:refersTo :monthlyUptimePercentage ] .
```

Listing 4. Compensation terms.

penalties are modeled using compensation terms. Unlike the previous use case, the guarantee of the compensation is a complex metric that specifies the formula used to compute the penalty instead of a numeric value. For all these elements, validity, evaluation and measuring intervals are modeled using the Time ontology. Specifically, the exclusion interval for the application of penalties is modeled by means of setting a validity interval for each guarantee that considers only the part of the contract for which penalties apply. Listing 6 shows some of these intervals defined in the call for tenders. Note that in this use case description we do not specify date and time descriptions of intervals, but only duration, since it is not describing a concrete bid but the maximum exclusion intervals defined in the call for tenders. Time Ontology does nonetheless provide support for specifying dates and intervals both in absolute and relative terms.

Finally, limits to monthly penalty costs and total penalty costs are defined as two guarantees whose guarantor is the Andalusian Health System that guarantee maximum values for the service properties that refer to these costs. Consequently, our model also supports that business entities involved in the SLA may have different liability roles depending on each agreement term.

5.3 Linked USDL Agreement Coverage Evaluation

In addition to the comprehensive use case validation previously discussed, we evaluate the coverage of Linked USDL Agreement against the comparison framework proposed in [6]. This comparison framework comprises 22 criteria covering the whole SLA lifecycle that were used to compare 14 different SLA and Service Contract Languages. Table 3

```
:highPriorityIssuesAvgResolutionTime
  a usdl-agreement:Metric ;
  qudt:unit <http://qudt.org/vocab/unit#Hour> ;
  usdl-agreement:hasExpression
    """SELECT ?s (AVG(?resDurationInHours) AS ?IO_05)
       WHERE {
         ?s sas-ict:hasIssue ?i .
         ?i sas-ict:hasPriority sas-ict:HighPriority .
         ?i sas-ict:hasResolutionTime ?resTime .
         ?resTime time:hasDurationDescription ?resDur .
         ?resDur time:hours ?resDurationInHours
       } GROUP BY ?s"""^^xsd:string ;
  usdl-agreement:hasMeasuringInterval :billingInterval .
```

Listing 5. Metric to compute average resolution time of high priority issues.

```
:contractInterval a time:Interval ;
  time:hasDurationDescription
    [ a time:DurationDescription ;
      time:years 2 ] .
:slaExclusionInterval a time:Interval ;
  time:intervalStartedBy :contractInterval ;
  time:intervalBefore :slaValidityInterval ;
  time:hasDurationDescription
    [ a time:DurationDescription ;
      time:weeks 8 ] .
:slaValidityInterval a time:Interval ;
  time:hasDurationDescription
    [ a time:DurationDescription ;
      time:years 1 ;
      time:weeks 44 ] .
```

Listing 6. Specification of contractual exclusion intervals.

summarizes the criteria and shows the evaluation results of Linked USDL Agreement. The last column showcases how many of the nine SLA languages analyzed fulfill each criteria.

In summary, Linked USDL Agreement fulfills 14 out of the 22 criteria. First, the variety of formalisms is high, using ontologies in our case. Functional and quality terms can be both described in Linked USDL Agreement through Linked USDL Core's `ServiceOffering` and `Guarantee` terms, respectively. Reusability is achieved thanks to the Linked Data approach used to design our model. Alternative service levels can be specified in Linked USDL Agreement through different `ServiceOfferings` for the same service, or by means of preconditions. The `MetricProvider` business role played by an involved entity, and `hasMeasuringInterval` property of `Metric` fulfill metric providers and metric schedule criteria, respectively. The condition evaluator also exists in our model as a business role. Qualifying conditions are equivalent to preconditions of the `AgreementTerms`. Liability roles model obliged parties for each `AgreementTerm`. Using properties like `hasEvaluationInterval` and `hasValidityInterval` we can model the assessment schedule of an SLO and validity periods for each `AgreementTerm`. Both penalties and rewards are modeled as `Compensations` in Linked USDL Agreement, and they can be linked using the property `hasCompensation` to an SLO (a `Guarantee` in our model). Finally, the SLA validity period can be also expressed using the `hasValidityInterval` property, or rather using the `validFrom` and `validThrough` properties of a `ServiceOffering`.

Note that metric definitions are not directly supported, since Linked USDL Agreement relies on external vocabularies to define service property metrics. As a result, our model is not coupled with particular mechanisms to compute metrics, since that is a responsibility of each metric provider and depends on each service scenario. Concerning composability, a single `ServiceOffering` that bundles several single services can be considered a particular case of a composite service, enabling the definition of agreement terms guaranteed over specific services included in the offering.

Regarding the rest of the criteria, they are not directly covered by Linked USDL Agreement because they are not present in most real-world SLAs we have found in our analysis [7]. Specifically, the ability to express soft constraints is not supported since most SLOs are formalized as hard requirements; the negotiation since most SLAs are take-it-or-leave-it offers without room for negotiation; and the ability to express recovery and settlement actions since SLAs generally only specify penalties. Moreover, the number of proposals actually fulfilling these criteria are no more than two except for recovery actions that are provided by four proposals. Consequently, they are only useful in very specific scenarios.

Nevertheless, Linked USDL Agreement can be easily extended to cope with additional features that may be needed in some scenarios, thanks to its design approach embracing Linked Data principles. For example, a negotiation-related extension could be designed by extending the

TABLE 3
Linked USDL Agreement Evaluation According to the Comparison Framework from [6]

Criteria	Description	Evaluation	Proposals
Formalism	The language's formalism	Ontologies	Various
Coverage	The ability to express functional and quality terms	[y,y]	2 [y,y]
Reusability	The ability to reuse parts of the SLA	yes	7 yes, 2 part.
Composability	The ability to represent SLAs for composite services	partial	1 good, 4 fair
Metric definition	The ability to define quality metrics	external	5
Alternatives	The ability to express alternative service levels	yes	7 impl.
Soft constraints	The ability to express soft SLOs	no	2
Matchmaking Metric	Definition of how to compare SLAs	no	2
Meta-Negotiation	The ability to represent information about the negotiation process	no	1 good, 2 fair
Negotiability	The ability to define which parts of the SLA are negotiable	no	2 part.
Metric Provider	The ability to define the party responsible for producing metric's measurements	yes	4
Metric Schedule	The ability to define the measurement frequency of a metric	yes	4
Condition Evaluator	The ability to define the party responsible for SLO evaluation	yes	2
Qualifying Condition	The ability to define conditions that must hold in order to assess an SLO	yes	2
Obliged	The ability to express the party in charge of delivering what is guaranteed in an SLO	yes	7
Assessment Schedule	The ability to express the assessment frequency of an SLO	yes	3
Validity Period	The ability to express the time period in which the SLO is guaranteed	yes	4
Recovery Actions	The ability to express corrective actions to be carried out when an SLO is violated	no	4
Penalties	The ability to express penalties incurred when one party violates its guarantees	SLO	3 SL, 2 SLO
Rewards	The ability to express rewards incurred when one party exceeds its guarantees	SLO	1 SL, 2 SLO
Settlement Actions	The ability to express actions concerning the final SLA outcome	no	2
SLA Validity Period	The ability to express the period where an SLA is valid	yes	5

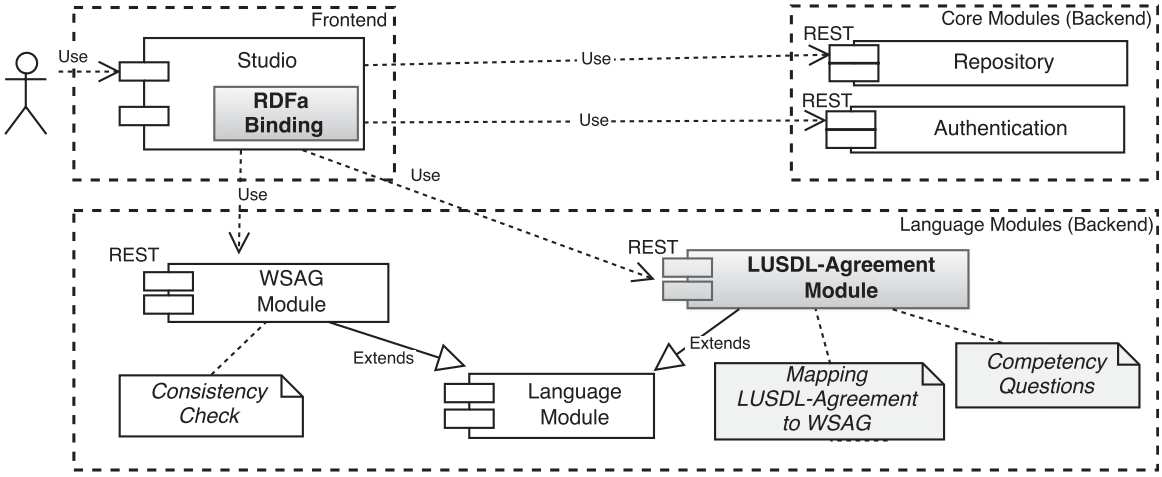

Fig. 4. Tooling architecture.

`ServiceOffering` with information about the negotiation process and adding negotiability properties to `Guarantee` terms.

In conclusion, Linked USDL Agreement sufficiently supports the definition of the most common features required to describe an SLA according to the comparison framework proposed in [6], and it actually performs as one of the best 14 SLA and service contract languages analyzed in the above mentioned comparison. Furthermore, using formal semantics to describe SLAs is a major advantage of our model in contrast to existing approaches, enabling automation of the SLA lifecycle by sharing machine-understandable SLA definitions on the Web.

6 Linked USDL Agreement Tooling

Describing an SLA using Linked USDL Agreement (as with any other formal language) can be a challenging task. Being a manual activity, there is a risk of introducing errors that, depending on the complexity of the agreement, can be very high. Furthermore, since SLAs specify rights and responsibilities of the stakeholders that could lead to compensations, their statements include sensitive information that should be carefully modeled. In this context, inconsistencies and other conflicts between terms of the SLAs constitute a major issue to be avoided in order to prevent misunderstandings or unexpected situations. Moreover, the usage of formal languages represents an important barrier for non-technical users that are used to natural language with an appropriate human-oriented structure and syntax.

In order to face these challenges we provide a convenient tool[22] for the formal modeling and consistency checking of SLAs. Additionally, we provide an intuitive RDFa-based solution to define natural language views of the document that are bound to the formal Linked USDL Agreement in order to i) synchronize the natural language description and the machine readable serialization; and ii) to help non-technical users in producing rich descriptions suitable both for humans and for machines in an easy manner.

In the following sections, we describe the different elements developed: first, we introduce the architecture of the solution; second we present the different analysis paradigms used for both validity checking and competency questions; finally, we present the RDFa extension we have developed.

6.1 Architecture

The tool has been developed within the context of the IDEAS framework that provides a generic on-line development environment for domain-specific languages (DSL).

Specifically, as shown in Fig. 4, the IDEAS framework is composed of an user front-end that provides the common functionality of on-line development environments such as authentication, file management and console; also, there is a generic client-based editor that can be parametrized with different grammars and syntaxes. On the back-end, IDEAS proposes a standardized module system that is based upon REST interfaces. Each module is associated with a particular language and it is launched when a particular file is loaded based on its type; the language is represented as a model that can be serialized in different formats; consequently, once the file is loaded, the editor supports the change between different views in each of the formats of the specific language. The structure of a module is comprised of two sets of operations: (i) language management operations that provide syntax checking and the marshaling and unmarshaling of the different formats and (ii) analysis operations that provide specific functionality to extract information over the document loaded.

In Fig. 4, highlighted elements represent the specific extension of the IDEAS platform for the current work. In particular, this extension comprises three parts:

- A new IDEAS module has been created to support editing and analyzing Linked USDL Agreement, using the Turtle RDF syntax. This module provides a set of operations based on our competency questions transformed into generic SPARQL queries that can be dynamically executed over any instance of documents in the platform.
- A binding extension to integrate RDFa enriched descriptions in natural language with a particular Linked USDL Agreement document. This binding provides an entry level mechanism for non

22. A demo of the tool presented is available at http://www.isa.us.es/IDEAS/Linked_USDL_Agreement

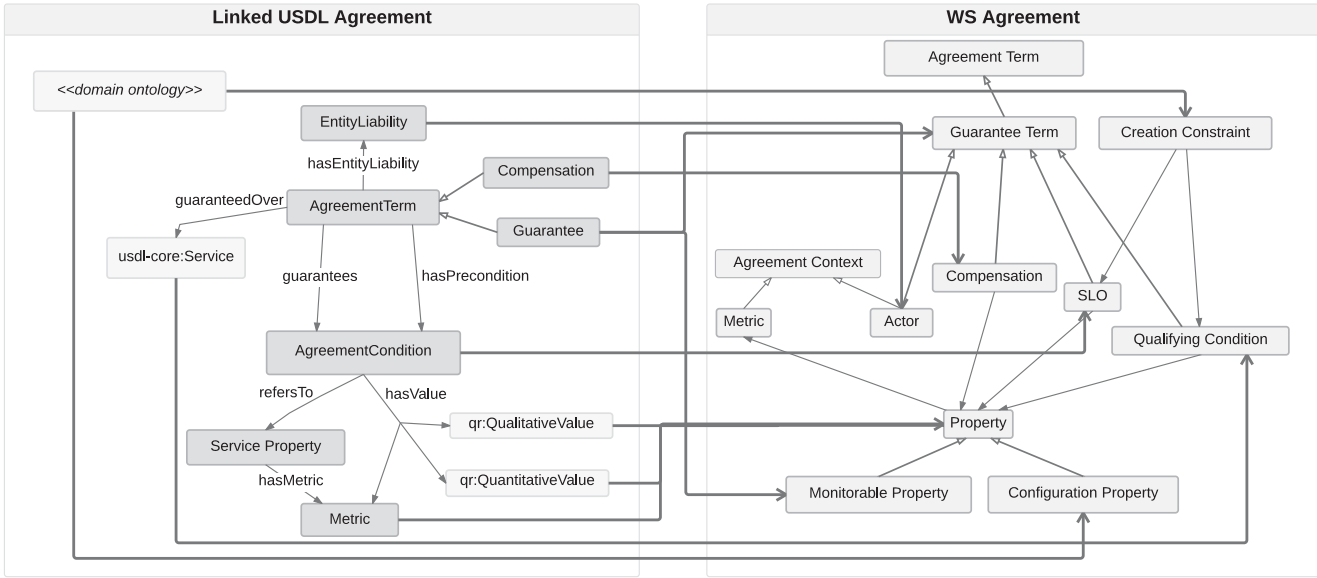

Fig. 5. Mapping from linked USDL Agreement to WS-Agreement.

technical uses to handle Linked USDL Agreement documents.

- In order to boost the applicability, we have developed a transformation from Linked USDL Agreement to the WS-Agreement specification since there is a fully functional IDEAS module already developed in the IDEAS framework. This module integration provides the possibility of a wider range of validity analysis.

6.2 Analysis of Linked USDL Agreement Documents

The tooling developed provides two different perspectives of analysis: on the one hand a set of validity operations that helps the user to create a correct document avoiding different types of potential conflicts; on the one other hand, the tool provides a list of analysis operations to answer the competency questions enumerated in Section 3.2.3.

Validity analysis is based on an analysis of the constraints defined in agreement conditions. In order to address this goal, we reuse the constraint programming based technique presented in [14]; specifically, we have developed a transformation from Linked USDL Agreement to a WS-Agreement template that directly maps those constraints. This solution enables the exploitation of our existing infrastructure for analyzing the validity of constraints, while providing support for importing legacy agreements in WS-Agreement. In Fig. 5, we present the different relationships between the concepts of Linked USDL Agreement and WS-Agreement; based on this conceptual mapping, the transformation is carried out in three stages as follows:

1) Linked USDL Agreement guarantee terms are transformed into WS-Agreement guarantee terms, where guarantees, preconditions and compensations are mapped to SLOs, qualifying conditions and penalties or rewards in business value lists, respectively.
2) Linked USDL Agreement service properties that are referred by agreement conditions are transformed into WS-Agreement service properties as variables.
3) Service properties that are included in service offerings are transformed into properties in the service description terms of WS-Agreement, and their concrete values are transformed into creation constraints for those service description terms.

In order to check the validity of an SLA, once the WS-Agreement document is generated, the IDEAS WSAG Module engine performs a transformation over a set of constraints models that are fed into a CSP solver (cf. [14] for details on this module). Specifically, in the current tool, document validity is decomposed in three different analysis that look for specific problems:

- *Global Inconsistencies*. This error represents a conflict between two constraints specified in the different terms of the document. This conflict can arise if some agreement conditions contradict each other, for instance when they state that *memory should be less than 100 and greater that 120*.
- *Conditional Inconsistencies*. This conflict is a special kind of inconsistency that depends on the precondition established. As a consequence, this problem only appears in specific selection of properties, though, depending on the complexity of the agreement, this case can be usual. As an example we can have two different terms such as *if the size is large the memory should be 10 GB* and *if the region is eu-central-1 the memory should be less than 5 GB*; in this example, a potential configuration of the agreement with both *region eu-central-1* and *size large* represents a conditional inconsistency.
- *Dead Terms*. This case represents an inconsistency of a particular agreement term with the precondition of another term. In such situation, it is not possible to meet the conditions required and therefore the term is never guaranteed. For instance, an agreement term stating that *memory should be greater than 10 GB* flags a term whose condition is *if the memory is less than 5 GB the region should be eu-central-1* as a dead term.

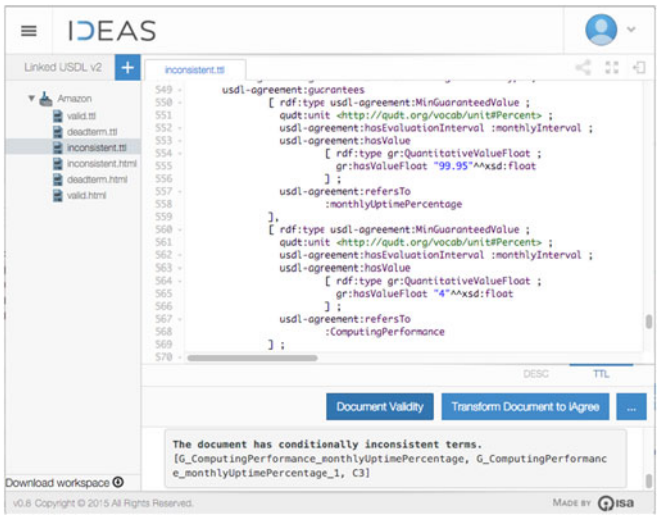

Fig. 6. Screenshot of validity analysis.

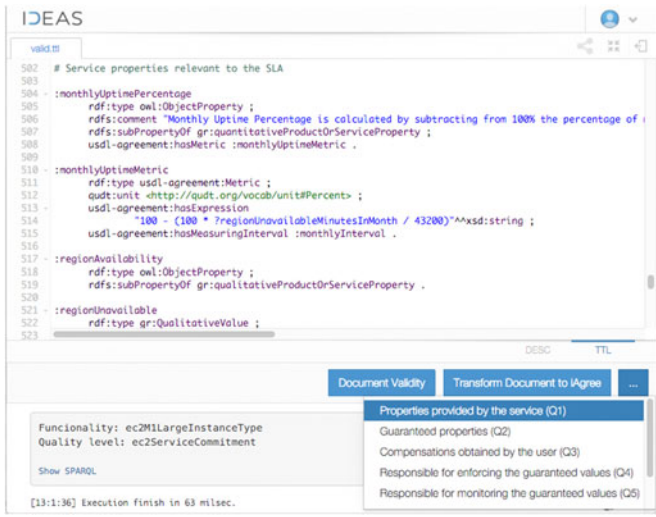

Fig. 7. Screenshot of competency questions support.

Fig. 6 shows our tool performing a validity check over the document. Note that this operation internally transform the Linked USDL Agreement description into a WS-Agreement document, transparently to the user. However, our tool also provides the possibility to explicitly generate the WS-Agreement document into the workspace.

Question answering over SLA documents is supported by means of SPARQL queries. These queries can be extended to provide a wide range of analysis operations. In Fig. 7 a screenshot shows the execution of a specific competency question operation, where users can actually see the SPARQL query that is executed over the document and possibly fine-tune or tweak it as they deem appropriate.

6.3 RDFa Support

Directly using formal languages for modeling concrete SLAs in terms of Linked USDL Agreement requires a level of knowledge and expertise of advanced Information Technologies that is not often available to many users. Bridging this gap is a fundamental requirement for the general adoption and use of the model. In order to overcome this challenge, the tool provides a binding mechanism based on the RDFa[23] standard that allows tying natural language documents to their formal representation in Linked USDL Agreement. In a nutshell, RDFa provides a set of markup attributes to be able to attach structured metadata to HTML pages in a way such that machines can directly glean and process that information. On the basis of RDFa Lite (which is a subset of the RDFa standard that provides five simple attributes to include information about the vocabularies, concepts, their types and properties), IDEAS allows users to directly bind Linked USDL Agreement definitions to Web pages. Doing so simplifies the generation of structured descriptions, ensures that natural language SLA definitions are consistent with their corresponding machine processable descriptions, and it seamlessly enables the enrichment of existing Web pages which are currently the main source of SLA descriptions on the Web.

In order to take advantage of the RDFa support, an initial description of the agreement should be developed in HTML; based on this description, a binding process can be started with our tool. Specifically, the tool provides an RDFa assistant that guides the expert into the definition of bindings; this binding process comprise three different stages:

1) A selection over the Linked USDL Agreement document is chosen; the user can specify multiple selections and it is possible to define composed bindings over previous atomic ones.
2) A selection of the fragment of description is specified; this fragment will be linked with the formal fragment selected in the previous stage.
3) A configuration of the binding is established. In this stage, all the different attributes of RDFa Lite can be specified.

Fig. 8, shows the final stage of an specific binding where a property is set for the concept *cloud:hasComputingPerformance* in order to be linked into a *span* element of the HTML description. In addition to the assistant, the tool also provides a list of management features to delete and update the different bindings in a document.

In case a user opens a document with a description that includes binding information, the tool first shows

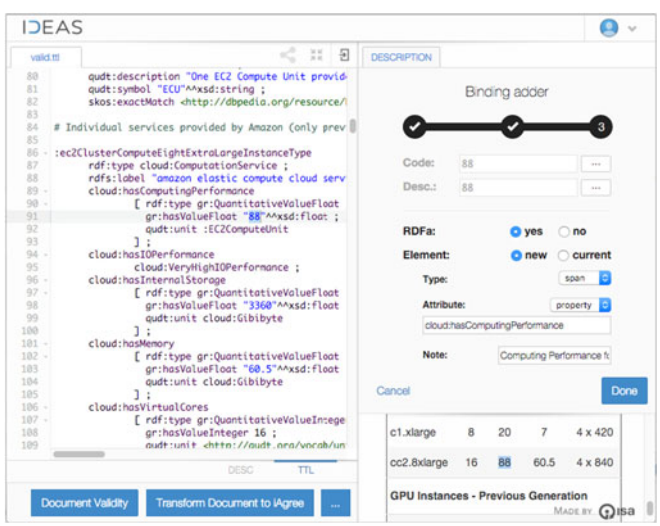

Fig. 8. Screenshot of RDFa binding support.

23. http://www.w3.org/TR/xhtml-rdfa-primer/

the natural language description, so the user does not need to understand the underlying Linked USDL Agreement model to start using the document. If necessary, the user can change to the RDF view in Turtle syntax and the tool will split the interface to show both the description in natural language and the Linked USDL Agreement model with the different bindings highlighted. Noteworthy, the tool provides a bi-directional binding that enforces the consistency between the two views of the document. Any changes to the natural language description get reflected on the Linked USDL Agreement model and vice versa.

7 Conclusion and Future Work

In this article we describe Linked USDL Agreement, an extension to the Linked USDL service description family, that can capture agreement terms, business aspects, liability, compensations, and time constraints. This approach contrasts with previous specifications such as WS-Agreement, WSLA, and SLA* that were mainly developed to address the technical aspects of Web services. Specifically, Linked USDL Agreement is designed to be used to establish and share agreements among customers and providers that seek to perform automated service trading in the context of the Web.

The evaluation process has been carried out from different perspectives. First, we have evaluated our model expressiveness to describe real computational services such as the AWS Elastic Computing Cloud (EC2), as well as the case of non-computational services that are present in a business process outsourcing scenario. Second, we describe how our proposal addresses the lifecycle of an SLA compared to the related work, with a special focus on common features that are present in real SLAs. Moreover, we explore how the knowledge captured by our model can be managed by tools to conduct analysis operations such as a validity checking. Finally, our implemented tool provides annotation features that can bind natural language documents describing SLAs to the underlying Linked USDL Agreement formal descriptions, fostering the adoption and usage of our model by non-experts.

Concerning future work, we are currently developing a prototype to support the construction of potential service marketplaces that could provision Linked USDL services in an automated way to consumers based on their requirements and preferences [20], while addressing heterogeneity issues and formalizing contracts using Linked USDL Agreement. Moreover, we are integrating our proposal with an SLA monitoring solution in order to automatically detect violations on service level objectives that may trigger compensations and notify corresponding customers.

Acknowledgments

Authors would like to thank Juan Luis de la Fuente and Felipe Serafim for the development of the tooling. This work has been partially supported by the European Commission (FEDER), the Spanish and the Andalusian R&D&I programmes (grants P12-TIC-1867, TIN2012-32273, TIC-5906, TIN2015-70560-R, COMPOSE FP7-ICT 317862).

References

[1] C. Pedrinaci, J. Cardoso, and T. Leidig, "Linked USDL: A vocabulary for web-scale service trading," in *Proc. 11th Int. Extended Semantic Web Conf.*, 2014, pp. 68–82.

[2] H. Akkermans, Z. Baida, J. Gordijn, N. Peña, A. Altuna, and I. Laresgoiti, "Value Webs: Using ontologies to bundle real-world services," *IEEE Intell. Syst.*, vol. 19, no. 4, pp. 57–66, Jul./Aug. 2004.

[3] J. Cardoso, A. Barros, N. May, and U. Kylau, "Towards a unified service description language for the internet of services: Requirements and first developments," in *Proc. IEEE Int. Conf. Services Comput.*, 2010, pp. 602–609.

[4] J. Cardoso, T. Binz, U. Breitenbücher, O. Kopp, and F. Leymann, "Cloud computing automation: Integrating USDL and TOSCA," in *Proc. 25th Int. Conf. Adv. Inf. Syst. Eng.*, 2013, pp. 1–16.

[5] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data—The story so far," *Int. J. Semantic Web Inf. Syst.*, vol. 5, no. 3, pp. 1–22, 2009.

[6] K. Kritikos, et al., "A survey on service quality description," *ACM Comput. Surveys*, vol. 46, no. 1, pp. 1–58, 2013.

[7] J. M. García, C. Pedrinaci, M. Resinas, J. Cardoso, P. Fernandez, and A. Ruiz-Cortés, "Linked USDL agreement: Effectively sharing semantic service level agreements on the Web," in *Proc. IEEE Int. Conf. Web Services*, 2015, pp. 137–144.

[8] J. Cardoso, R. Lopes, and G. Poels, *Service Systems—Concepts, Modeling, and Programming*. Berlin, Germany: Springer, 2014.

[9] J. Cardoso and C. Pedrinaci, "Evolution and overview of linked USDL," in *Exploring Services Science*, H. Novoa and M. Dragoicea, Eds. Switzerland: Springer Int. Publishing, 2015, pp. 50–64.

[10] I. Arampatzis, S. Veloudis, and I. Paraskakis, "Linked USDL business policy specifications as WS-agreement templates," in *Advances in Service-Oriented and Cloud Computing*, G. Ortiz and C. Tran, Eds. Switzerland: Springer Int. Publishing, 2015, pp. 221–232.

[11] C. C. Marquezan, A. Metzger, R. Franklin, and K. Pohl, "Runtime management of multi-level SLAs for transport and logistics services," in *Proc. 12th Int. Conf. Service-Oriented Comput.*, 2014, pp. 560–574.

[12] A. Dan, et al., "Web services on demand: WSLA-driven automated management," *IBM Syst. J.*, vol. 43, no. 1, pp. 136–158, 2004.

[13] A. Andrieux, et al., "Web services agreement specification (WS-Agreement)," Open Grid Forum, Muncie, IN, Tech. Rep. GFD-R.192, 2011. [Online]. Available: https://www.ggf.org/documents/GFD.192.pdf

[14] C. Müller, M. Resinas, and A. Ruiz-Cortés, "Automated analysis of conflicts in WS-Agreement," *IEEE Trans. Services Comput.*, vol. 7, no. 4, pp. 530–544, Oct.–Dec. 2014.

[15] M. Uschold and M. Gruninger, "Ontologies: Principles, methods and applications," *Knowl. Eng. Rev.*, vol. 11, no. 2, pp. 93–136, 1996.

[16] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: State-of-the-art and research challenges," *J. Internet Services Appl.*, vol. 1, no. 1, pp. 7–18, 2010.

[17] M. Hepp, "GoodRelations: An ontology for describing products and services offers on the Web," in *Knowledge Engineering: Practice and Patterns*. Berlin, Germany: Springer, 2008, pp. 329–346.

[18] C. Müller, O. Martín-Díaz, A. Ruiz-Cortés, M. Resinas, and P. Fernandez, "Improving temporal-awareness of WS-Agreement," in *Proc. 5th Int. Conf. Service-Oriented Comput.*, 2007, pp. 193–206.

[19] A. Gómez-Pérez, "Ontology evaluation," in *Handbook on Ontologies*, S. Staab and R. Studer, Eds. Berlin, Germany: Springer, 2004, pp. 251–274.

[20] J. M. García, M. Junghans, D. Ruiz, S. Agarwal, and A. Ruiz-Cortés, "Integrating Semantic Web services ranking mechanisms using a common preference model," *Knowl.-Based Syst.*, vol. 49, pp. 22–36, 2013.